Gesture Recognition Using Histogram of Optical Flow

Ruochen Wen

2017 June

Abstract

In the field of Computer Vision, Gesture Recognition is kind of crucial problem. What differ video classification from normal image classification is that the enormous amount number of video data can not be ignored, because those complex data could lead to significant decline of computational efficiency, Therefore, this article mainly focus on how to obtain a video classification with both accuracy and efficiency in the meanwhile. In order to create a n ideal video classification system, the article use an improved speed-up Bag-of-Words model as basic pipeline. In each part of the pipeline, we apply and evaluate various strategies. In particular, in the step of feature extraction, we create a type of fast information feature descriptor for video, which is called Histogram of Optical Flow. Besides, we try to modify frame sampling rate of video, aiming to reduce calculation. In the process of creating features, we use sampling rate which is same to the size of a block. In this way, each block could be used repeatedly and the calculation will be reduced. When building visual word vocabulary and using SVM for classification, we use different methods to find a best performance of our system. As a final result, we get a trade-off between efficiency and accuracy of our gesture recognition system.

Key words: Optical Flow, Feature Extraction, Bag-of-Words, Gesture Recognition

Contents

1.Introduction

1.1 The main challenge in gesture recognition

Today, one of the main challenges of gesture recognition is the processing of large numbers of data sets. Because with the rapid development of the Internet, more and more data will appear in the form of video, which led to the already just for the image data, some applications have been unable to solve some problems. Because, although some theoretical methods have a good performance in the field of static images, since the video data contains complex information and the information is in the process of changing, resulting in the calculation of the processing of video data is quite large and is quite Therefore, when these methods are applied to the posture recognition problem composed of video, the result is not satisfactory, and there are some specific and new problems. Therefore, in order to solve the special problems in the attitude recognition, the face of the system needs to deal with a large number and its number is still growing in the video data, to explore and find those who do not enter the accuracy of the performance of the method, There is also a significant increase in the speed of the method is very necessary.

1.2 Research status at home and abroad

In this section, we mainly introduce the characteristics of the description, introduce their domestic and international development.

Today, the most commonly used local descriptors (on time and space spans) are descriptive features based on scale-invariant feature transformations. The establishment process is as follows: Each local video sequence will be divided into different partitions, in each partition we will partition all the pixels in the calculation of the optical flow or the direction of the gradient results together, get a total of this partition A (flow of light or direction) to calculate the results. And the final descriptor

is the adjacent, a specific number of partition results through a certain way to get together. Respectively in 2006 and 2008, the gradient direction histogram descriptor and the optical flow histogram descriptor on the two-dimensional plane are proposed. In addition, Dalal also presents a descriptor obtained by calculating the optical flow variation: the motion boundary histogram feature. Scovanner and Kläser, respectively, in 2007 and 2008, have proposed on the basis of two-dimensional plane, while the time dimension also established the direction gradient descriptor theory, which has been three-dimensional The descriptor. In 2013, the three-dimensional spatial description sub-theory was extended in terms of image channels. But then found that the fact that the proposed descriptor is not better than the direction of the histogram description sub-effect is better.

We can see that some points of interest selection theory and some time-space feature descriptors are evaluated. He found that intensive sampling theory in general would be better than the way of selecting points of interest, especially for some of the more complex and difficult to identify the data set.

Recently, the theory of video-intensive trajectory is proposed. In this theory he suggests that local video information moves over time, and the result is that, even if the time changes, the above-mentioned local information is still trying to remain in the same part of the target object. In addition, they also use the flow of light rather than light flow as the basis for characterization. Their work in the basic gradient direction histogram, optical flow histogram and action boundary histogram characterization descriptor made some improvements. However, combining their new tracing descriptive characteristics with traditional gradient directions, optical flow histograms, and motion boundary histogram characterization descriptors will actually result in a better effect than using a tracking feature descriptor alone.

It is proposed to use the integral graph to effectively calculate the characteristics of the accelerated file in the static image. And put forward some in the image classification problem, to accelerate the classification of word bag classification of some of the theory and their performance made a very detailed assessment.

In recent years, in the application of the word pocket model, the Fisher vector has

shown a better performance than the standard classification theory (such as k-means mean clustering). And the recently proposed VLAD descriptor local descriptor aggregation is seen as a non-probable version of the Fisher's vector.

1.2 The main content of this article

The main objective of this paper is to obtain a posture recognition system that achieves a relative optimal result (or a balance between the two) with the efficiency of the calculation and the accuracy of the results. Around this purpose, we used a fast word bag model as the basis of the pipeline, at different stages of the program, the use of image feature extraction, visual vocabulary classification and classification, video data classification and identification of a variety of methods, and They were evaluated and compared separately.

First of all, according to some of the characteristics of video data, we have adopted a number of methods, we hope to improve the speed of the system: For example, we have how to focus on the video data sampling method to do a certain study: We try to change the sampling rate for video frames Improve the efficiency of the method.

Secondly, since the optical flow histogram feature is used as the descriptor, we will also pay attention to the selection of the calculation method of the optical flow. We use and evaluate the different optical flow algorithm. At the same time, we try to build a dense, accelerated optical flow histogram descriptor on the basis of the underlying optical flow histogram descriptor.

Furthermore, on the basis of the existing phonetic model, we try to find and evaluate a combination of the most suitable for video classification to achieve the above-mentioned automatic recognition system with both accuracy and computational efficiency.

In addition, we used and evaluated the effect of using Fisher's vector theory and k-mean, hierarchical k-means hierarchical clustering in video classification.

Finally, we use different classifiers to match the previous steps to achieve the best video classification results.

2 Bag-of-Word Model

In the field of attitude recognition based on visual features, the emergence of a model occupies the mainstream: the word pocket model. The word pocket model has been proven in a number of experiments that he is the most effective strategy for general classification strategies. We can derive this conclusion from the excellent performance of the various types of mainstream evaluation systems over the past few years, such as the TRECVID advanced feature extraction task in video data. In these tasks, conceptual descriptive features can be detected and assigned to different categories such as chairs, cats, cars, and the like. Based on the excellent performance of the word pocket model, this text is based on this classic model as our basic design pattern. But the basis of the performance of the word pocket model still can not achieve our goal, its huge amount of calculation can not be ignored defects. So on this basis, we have improved the classic word bag model, so that it is not only in the efficiency and accuracy are reached a higher level, this strategy we will be detailed after the introduction.

2.1 Introduction of Bag - of - Word Model

The name of the phonetic model is actually very vivid, that is, with a group of "words" to express our extracted features. In the beginning, there is actually a question about the text retrieval problem. The specific process is as follows: First, a dictionary containing all the words in the training set text is constructed according to

the training set text. From another perspective, As a fixed-length vector, because the words in the text will be repeated, we will be able to each text in which the frequency of the word that, so that each text can be used to express the fixed-length vector, through training these data we will You can sort the text to be tested. Similarly, in the word pocket model, we use a similar strategy: in accordance with the data itself will be divided into training set and test set, we will train all the characteristics of the data set together, and apply a certain way to them The automatic classification (the total number of categories is determined by the user), the resulting classification center points represent each category, so that we get an array of size-specific information that contains the center point information to represent the visual word in our word pocket model. Then we will focus on the video of the video with the visual word in the dictionary, generate each video fixed-length vector used to train the support vector machine model. After completing the training, and then take the test set video for video classification. The basic word pocket model contains the four main steps: 1. Video extraction features. 2. Create a visual dictionary. 3. Generate the video histogram. 4. Video classification. The process is as follows:

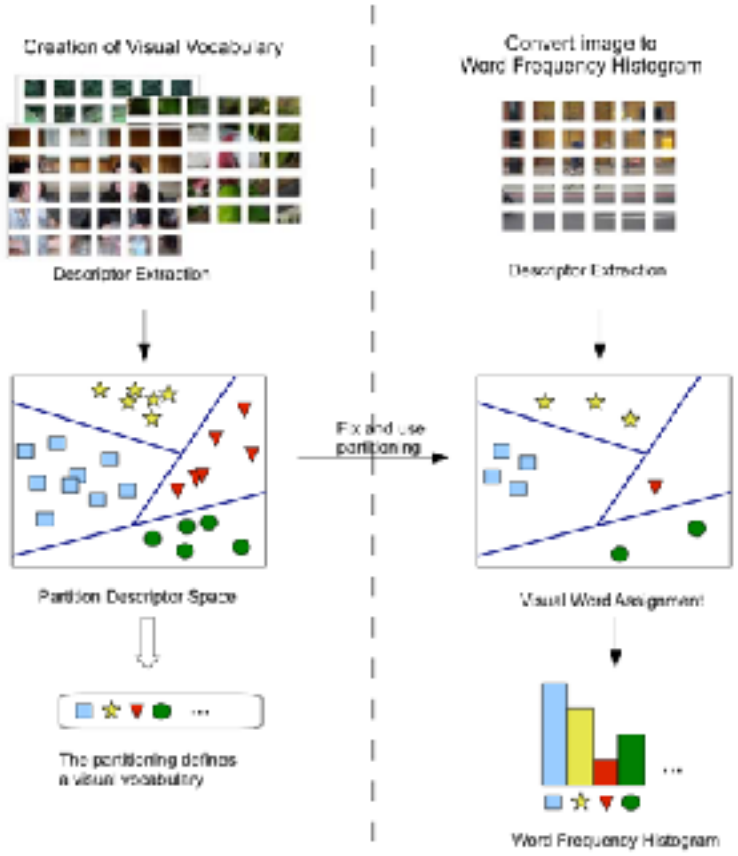

figure 2.1

Video Feature Extraction: General, before proceeding with other steps in the word pocket model. It is first necessary to extract features for each video in the training set to obtain information about the video and to use it for later identification. The specific process is described below, the video as a sequence of images, through the feature extraction strategy to obtain the characteristics of each video descriptor, so that we can each video with a m * n matrix, where m for this video The number of features, n is the characteristic dimension of the video (the general video feature dimension is the same). Identification problems arise from the origin to the present, resulting in a number of different feature extraction methods, which are widely used SURF, SIFT, HOG and HOG.

Create a visual dictionary and generate an image histogram: Once you have the characteristics of all the training set images, you can aggregate all of these features together and use some strategies to classify them. The determination of the number of visual words is a key point in the word pocket model, which affects the results of the classification after a large extent, and its number needs by the user according to the number of features and other factors, through repeated Experimental decision. After the visual dictionary is established, calculate the distance between each feature point of the image in the training center and the word of the visual dictionary, and classify the feature points to the nearest, Each of which can be represented by a fixed-length visual word vector, and the value of each dimension in the vector represents the number of features represented by the word. Then the vector is generated by the abscissa as the number of words, the ordinate for each word in the characteristics of the visual word frequency histogram, and this histogram will be used for the subsequent classification process.

Video classification: First of all, the general training set of each image with the corresponding histogram, they can be sent to the support vector machine classifier to start training classification model. Then, the same feature extraction method is used to get the feature descriptor of each video in the test set, and the video is represented by the words in the previously established visual dictionary, and the frequency

histogram of each video is generated correspondingly. Then, the histogram of the test set image processed above is given to the trained classifier model for processing, and the final video classification result is obtained.

2.2 Innovation and design

For the video data, the calculation is much larger than the image data, so the classification of video problems, accuracy is not the only measure, we also need to take into account the computational efficiency. Therefore, the main strategy of this paper is to improve the computational efficiency of video classification in the case of using the traditional phonetic model and reduce the sacrifice of accuracy as much as possible.

In order to achieve the above purpose, this paper mainly proposes the following design scheme: In each step of the word pocket model, different methods are implemented and the results are compared to select the optimal combination.

Before extracting features, there is a problem - since each video contains a large number of frames, it results in a huge amount of computation. But in fact, this consumption is sometimes unnecessary, because the same video before and after the adjacent frame contains most of the same information. Taking this into account, we reduce the computational complexity by changing the sampling frequency of the image by using a different step size to sample a video in time. In addition, since hof is built from subvolumes in the video, and they can be reused by different descriptors, the use of subdions in 3D space is used to reduce the computational complexity.

Feature extraction stage, we mainly use the optical flow histogram method as the feature extraction strategy, in order to further explore the different methods on the calculation accuracy and efficiency of the impact.

In the traditional method, a series of local visual descriptors are transformed into

fixed length vectors by k-means, and the local descriptors are classified by the nearest cluster method. In this paper, based on the use of k-means, also used hierarchical k-means and fisher vector. Hk has the advantage of fast, and fisher classification effect is far better than k.

Image classification stage, this paper uses the linear SVM, the use of histogram cross-core SVM, and the establishment of different stages of visual dictionary combination, trying to find the best combination of programs.

3.Feature Extraction

In this chapter we will mainly introduce the feature extraction process, we will introduce a posture recognition for the outstanding performance of the descriptor. We will mainly from the theoretical and practical aspects of the description of the

description of the utility.

In this paper, we mainly use the optical flow histogram as a descriptor for attitude recognition. First of all, we will briefly introduce the principle of light flow and classification, as well as the resulting optical flow field into a histogram process. In particular, we will introduce a special method of efficiently associating pixel optical flow histogram information with a high computational efficiency while covering a comprehensive descriptor.

In fact, there are many ways to extract features, which are more common with scale invariant features, gradient direction histogram features and optical flow histogram features.

3.1 Optical flow method

The first presentation of "light flow" dates back to 1950, first proposed by gibson. The reason for this is as follows: For the movement of moving objects in three-dimensional space is difficult to represent in the two-dimensional plane, gibson pointed out that we can be three-dimensional space moving objects projected into two-dimensional space, with the image of pixels To represent these objects. Correspondingly, the optical flow is the velocity at which the pixels of each frame are presented in the image. Thus, the optical flow of all the pixels in an image is linked together, and we get the optical flow field of the frame, which represents the distribution of the pixel light flow. Since the target object we are concerned with is moving continuously, the optical flow field generated by each two-frame sequence is different; that is, the optical flow field is constantly changing, and it reflects the target object in the time dimension Of the movement, and these optical flow field will form a constantly changing visual flow. Thus, the concept of "light flow" is obtained.

After this theory has been put forward, in order to achieve and apply the concept of "optical flow" to practical engineering, many researchers try to use a variety of different strategies to calculate optical flow, in which there are two more

commonly used methods: Lucas -Kanade as the representative of the sparse optical flow and Horn-Schunck as the representative of the dense optical flow.

3.2 Horn-Shrunk dense optical flow

The dense optical flow algorithm is characterized by the fact that the pixel-level optical flow in the image is involved, thus resulting in a great deal of computational work. The dense optical flow algorithm mainly has Horn-Schunck method and block matching method, the latter is now less applied. In this article we also use the law, so here we mainly introduce the law:

First, in order to calculate the optical flow, and put forward two hypotheses:

In the adjacent area, the direction and size of the moving speed of the target object are almost unchanged.

Grayscale changes in the moment is very small.

At the same time, they use the following equation as a constraint:

The calculation formula of the optical flow field is:

$$\iint \{\varepsilon(x,y)\cdot(I_x u + I_y v + I_t)^2 + \alpha^2[(\partial u/\partial x)^2 + (\partial y/\partial x)^2 + (\partial v/\partial x)^2]\}dxdy = min \qquad (3.1)$$

3.3 Lucas-Kanade sparse optical flow

Unlike dense optical flow, sparse optical flow only needs to deal with some pixels in the image, involving a smaller range, so the calculation is small. In order to obtain the corresponding feature points in the two images, this algorithm includes the following three steps, which are: 1. Construct the image layer pyramid. 2. Tracking on the basis of the pyramid. 3. Iterative calculations. That is, after all the image layers are obtained and initialized, the optical flow and the affine transformation matrix are iteratively calculated from the lowest image layer of the pixel. Finally, the result of the original image layer is the required optical flow and affine transformation matrix, In this process we strive to achieve every step of the error as small as possible.

3.4 Establish optical flow histogram

For the image action sequence, the optical flow is a good description of the characteristics. But untreated optical flow can not be used directly. Because the number of pixels involved in a

person in the image will change, the size of the descriptor will also change over time. In addition, the optical flow calculation results are susceptible to background noise, size transformation and direction change.

In order to avoid these problems, it is a better choice to use the distribution of the optical flow as a characteristic to express the action change on the basis of calculating the optical flow. So the concept of optical flow histogram was presented.

First, the frame as the basic unit, where the pixel level of light flow. Then, each optical flow vector is converted according to its size and magnitude of the horizontal axis. Thus, all optical flow vectors can be expressed as:

$$v = [x, y]^T$$

$$\theta = tan^{-1}(y/x)$$

$$-\pi/2 + \pi(b - 1/B) \leq -\pi/2 + \pi(b/B)$$

And they all pass the formula

$$\sqrt{x^2 + y^2}$$

added one by one to the b direction, where

$$1 \leq b \leq B$$

Since the angle of the optical flow vector is obtained by taking the minimum angle from the horizontal axis angle, this measure makes the result of the optical flow histogram not affected by the direction of the actual action, that is, they are mutually independent. After the completion of the above steps, the normalization of the optical histogram has made the scale does not deform the characteristics of the scale. At the same time, the tiny noise flow also has little effect on the optical flow histogram, so that the feature descriptor has a good ability to resist noise. The number B in the direction mentioned above is generally variable

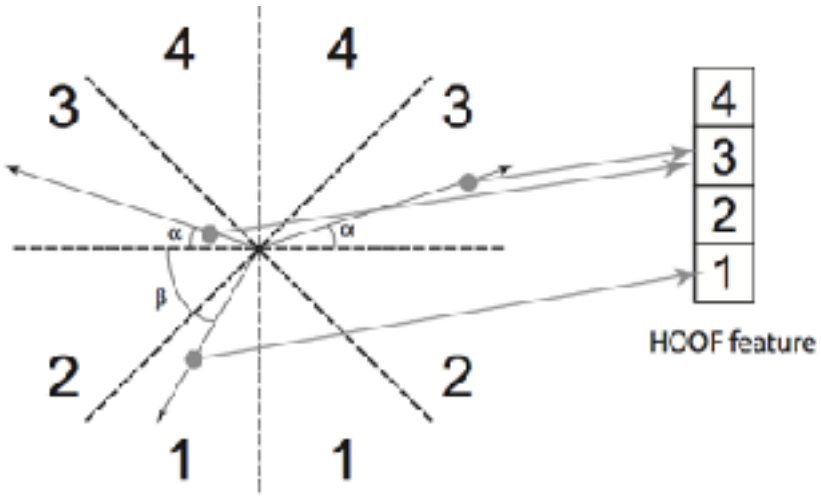

figure 3.1

3.4 Fast optical flow histogram descriptor

In this chapter, we will focus on a fast optical flow histogram descriptor we built. Unlike the feature information in the image, we made the corresponding improvement according to the characteristics of the video. In order to completely extract the video stream histogram features, the main need to complete the following steps:

Before we start feature extraction, we have a problem to consider: There are many frames in the video. To calculate the flow of each frame, it is conceivable that the computation is huge. At the same time, however, the information contained in the adjacent video sequence is mostly duplicated. Thus, some frames in the video can be extracted by frame sampling instead of calculating the number of frames, such as every two, three, or six frames, and the sample rate is According to the video itself, determined by experiment. By taking this approach, the computational efficiency of the feature extraction phase system is significantly increased.

After completing the preprocessing step above, you can start the feature extraction. First, it is necessary to calculate the optical flow of each frame in the frame in units of frames. In the first step, it is necessary to calculate the optical flow

displacement vector in the horizontal direction and the vertical direction in the two-dimensional image of each frame. Then, on this basis, in order to establish the optical flow histogram, we need to quantize the amplitude of the optical flow obtained for each pixel, and generally o = 8. In this way, the optical flow of each pixel can be represented by a vector whose direction represents the instantaneous motion of the pixel and its absolute value represents the instantaneous velocity of the pixel. The next step is to combine the results of each pixel, according to the time and space set in the block together, so we get each piece of light flow results; this process we will be detailed in the back , Which uses a coefficient matrix to calculate. The last step is to associate the adjacent blocks according to the pre-set size, and the resulting optical flow histogram results together to obtain the optical flow histogram characterization of the video.

Among them, the optical flow calculation part, the default Horn-Shrunk method used to calculate the optical flow of each pixel, in addition to the Lucas-Kanade method used to calculate the optical flow.

In this paper, the optical flow histogram characterization descriptor is constructed by using block as the basic unit. So, how to select the video in the block is a feature extraction in a key point. In this paper, we choose to construct the feature descriptor with the same step size as a single block size, that is, we can block these characters can be reused by different feature descriptors, so that each block Information has been calculated in advance, then the calculation efficiency will be significantly improved. Once the result of each block is calculated, the descriptor can be obtained by concatenating a certain number of adjacent blocks of the optical flow histogram. In this paper, we use each frame to occupy 3 * 3 blocks in space, and occupy two blocks in time to form an optical flow histogram characterization descriptor. Of course, the size of this feature descriptor is not necessary, it should be based on the specific data set to make the appropriate changes to get the best results. According to the above method, in addition to the video at the edge of the block, each block will be repeated by the different descriptors 18 times.

4. Building Visual Vocabulary

This chapter will focus on how to build the corresponding visual dictionary with the light stream histogram descriptor after getting the video.

The establishment of a visual dictionary consists of two phases: 1. Generating visual words. 2. Using visual words to represent video information features, and further use the image feature histogram to visually reflect this information.

4.1 Generate visual words

Generating visual words is a more abstract concept. It is the same dimension of each descriptor in each of the extracted videos, and in fact a video can produce a lot of descriptors and thus can not work concisely and concisely in the subsequent matching work, resulting in the use of a video descriptor After clustering, a number of feature vectors with a number less than the total number of descriptors are generated, but the dimensions are invariant, to effectively respond to the video information.

As can be seen from the above description, the choice of clustering is a key step in determining the visual word, and it also determines the speed of the process and the accuracy of the results. Therefore, it is important to choose a suitable word classification method. In this paper, we mainly introduce three methods: 1. k-means clustering 2. Hierarchical k-means hierarchical clustering 3. Fisher Vector.

4.2 k-means mean clustering

The goal of k-means mean clustering is to classify the data to be classified into k

clusters. First, the first step randomly selected k center points. The next step is to calculate the distance between each point to be classified and all the center points, where the center point closest to the result of the distance to be measured is the classification to be classified. And then calculate the distance between all points to be measured and the center point, to complete the first classification. On the basis of the completion of the first round of classification, according to the class has been divided into another class according to the algorithm to recalculate and select the center of each category and update, and then repeat the second step is once again classified. The above steps are repeated until the center point in the last category does not change, that is, the classification is completed and the final classification result is obtained. Among them, k-means mean clustering mainly faces the following problems: 1. The initial center point selection is random, leading to the final result due to the initial center point has a huge gap. 2. The number of selection k is not certain, the need for users based on data sets and experimental goals to determine. In general, the larger the value of k, the greater the amount of computation, but the results are more accurate (especially for large sample datasets).

4.3 Hierarchical k-means hierarchical clustering

As mentioned above, the main problem with k-means mean clustering is that the number of initial center numbers needs to be artificially determined, and finding a suitable k value is more difficult and requires multiple attempts; The initial center of the random selection of the results of the change is difficult to control. In view of these cases, Hierarchical k-means hierarchical clustering shows a better solution.

Hierarchical k-means Hierarchical clustering algorithm can be divided into two categories: "bottom-up" and "top-down".

Their processes are mainly used in the iterative algorithm. First introduce the "bottom-up" principle: Suppose we have a data set, which contains a point. First, we need to calculate the distance between all the points, get the two points with the smallest distance, divide them into groups, and denote them with a new point. In this way, the number of our data points is reduced to one. Then, repeat the above process

until the last only one classification. On the contrary, "top-down" is the opposite process.

It will eventually get a tree model results, which contain data depth, node, classification and other information.

4.4 Fisher Vector Fisher

As mentioned above, the main problem with k-means mean clustering is that the number of initial center numbers needs to be artificially determined, and finding a suitable k value is more difficult and requires multiple attempts; The initial center of the random selection of the results of the change is difficult to control. In view of these cases, Hierarchical k-means hierarchical clustering shows a better solution.

Hierarchical k-means There are two types of hierarchical clustering: "bottom-up" and "top-down". Their processes are mainly used in the iterative algorithm. First introduce the "bottom-up" principle: Suppose we have a data set, which contains a point. First, we need to calculate the distance between all the points, get the two points with the smallest distance, divide them into groups, and denote them with a new point. In this way, the number of our data points is reduced to one. Then, repeat the above process until the last only one classification. On the contrary, "top-down" is the opposite process.

Hierarchical k-means Hierarchical clustering The final result is a tree model that contains information such as depth, node, classification, and so on.

Fischer vector method is mainly based on the principle of Gaussian mixture distribution. First, we use the training set to get all the parameters of the Gaussian mixture distribution, that is to say, through the Gaussian mixture distribution to reflect the training set of video model, this step can be solved by EM method. This process can be further explained by setting the Gaussian mixture distribution model that has been known to have known parameters, substituting the video descriptor to be trained in step E, and obtaining a vector in the M step F, which is a correction value, indicating how it should be updated to apply to this data set. And this vector F is the Fisher Vector calculation results. The next step is to use the square root of the

absolute value to normalize the Fisher Vector. Although the Fisher Vector results are much more dimensionally than other methods, the linear classifier is more suitable for the Fisher Vector for subsequent use of the classifier, thus making up for the larger dimension in this step due to multidimensional The amount of calculation.

5.Video Classification

In this chapter, we mainly introduce the process of image classification. In the previous chapter we have obtained a histogram representation of all the images: that is, each image is finally represented by the corresponding fixed length vector.

In the classification process, we need to randomly divide all the images into three categories: training set, test set and verification set. However, in the actual situation, the verification set is used less, so we mainly divide the image into training set and test set in this paper. Among them, the number of their own data is determined by the user's own, but in general, the proposed training set and test set data ratio of 3: 2. Of course, there are other special methods such as cross validation, it will turn each data in turn as a test set to experiment.

After the above steps are completed, the support vector machine will be used in the classification. First, we need to enter the training set data, support the vector opportunity to learn the characteristics of each image in the training set, and then training the model. After the training model is obtained, the support vector machine will use the obtained model to predict the result of testing the concentrated image, and finally get the classification result. It should be noted that the user needs to adjust the parameters of the model to achieve the optimal results based on each training result in this process.

5.1 Introduction of Support Vector Machines

In many cases, we will need to attribute it according to the characteristics of a specific thing to classify it as one or more types. In computer science, we call this case a classification problem. SUPPORT VECTOR MEACHINE or support vector machine is a supervised learning model, which is a method of analyzing data and performing pattern recognition. The purpose is to automatically establish a series of rules that do not exist before the data to be classified Classify it.

By supervising learning we want to get the result that the system can infer the series of rules mentioned in the previous paragraph through the data that has already been marked. And no supervised learning means that the system has the ability to classify different targets into different clusters according to the similarity without knowing any known classification. The support vector machine belongs to the former, and we need to provide the tagged data All the classification, so that the system has the ability to follow our request to the new unmarked data assigned to the above categories.

6. Experiment

In this chapter, we will elaborate on our experimental process and some of the details. In addition, we will briefly introduce the database used in the experiment.

The entire experimental process is done through matlab, the main reason for choosing matlab is because it is easy to use the programming language, easy to implement the experimental process, and for image processing, computer vision has a rich toolbox. At the same time, because our program involves a large number of arrays and vector calculations and use, matlab is a better choice.

6.1 Experimental setup

In our experiments, we mainly carried out the following four assessments and comparisons: (1) We compared two methods of calculating the optical flow: horn and lucas (2) We compared three different ways of assigning visual words: k -means mean clustering, hierachical k-means hierarchical clustering and Fisher's vector method. (4) We used two different support vector machines: linear support vector machines and cross-histogram kernel support vector machines (4). We used two different support vector machines: linear support vector machines and cross-histogram kernel support vector machines.

For the unique characteristics of video data, that is, they are composed of multiple image sequences of this property. We used four different frame sampling rates, respectively, for every 1 frame, 2 frames, 3 frames and 6 frames, respectively, the feature extraction. At different frame sampling rates, we also change the size of the time dimension of the feature descriptor. The detailed contents are mentioned in the next paragraph.

In our process, feature extraction is a very important part, in this step we use a three-dimensional dense sampling optical flow histogram descriptor to represent the video information. In this section, we take every case of video sampling as an example. Wherein each of the descriptors consists of 3 * 3 * 2 blocks, where the third number "2" means that the two blocks are crossed across the video every block; further, each block is made up of 6 * 6 * 6 pixels (if every two frames are sampled, each block consists of 6 * 6 * 3 pixels). Therefore, in addition to the edge of the block, we will repeat each block 18 times. This re-use for the reduction of the system has a good effect, but also to ensure the integrity of the video information. In the calculation of optical flow, we default to the horn-schunck method, but at the same

time for comparison, we also use lucas algorithm to calculate, and the final classification of the two results were compared.

In the construction of visual dictionary this step, we first selected three different strategies to complete the visual word distribution, the establishment of. For the k-means mean clustering, we set the number of visual words to 100, the initial value of each central point. We try to avoid the use of randomly generated methods to reduce the gap between the results. For hierarchical k-means hierarchical clustering, we construct a tree structure containing two layers of depth and 10 branches per section (experiments show that when we use more levels, the classification effect will actually produce a significant decline) , The total count down there are 100 visual words. In this case we can fairly compare the results of the two methods. For the Fisher vector method, we used two common classification sizes: 64 and 256 clusters. Although the dimensions of the eigenvectors generated by the Fisher vector method are much more than those of the other two methods, we find that it is in good agreement with the linear support vector machine. The linear support vector machine works in A certain degree of balance due to the Fisher high-latitude generated by the calculation.

On the basis of our experimental results, we propose two best schemes for the requirements of real-time video classification and the demand for video classification accuracy.

6.2 Experimental results

6.2.1 The establishment of visual dictionary and the impact of video classification

For the results of the visual dictionary we used above, we use graphs to represent. It is clear from the table that the Fischer vector accuracy with a total of 256 clusters can be up to 90%, and the time it takes is 3.21 seconds. The results of the K-means mean clustering also have excellent accuracy and can be seen as a value of 88.6%. But it takes only 1.51 seconds, far less than the time required for the Fisher vector. This shows that

for video classification, the Fisher vector is more accurate than k-means clustering, but takes more time. The level of clustering is very short, in 0.39 seconds, but its accuracy is also slightly lower than the other two methods, at 85.9%.
And in the video categorization of this part of the results, we can see in the use of cross-core histogram support vector machine in the case, each processing a video used for 0.15 seconds. In the use of linear support vector machine, each processing a video used for about 0.01 seconds, while the linear support vector machine is mainly used in the previous step in combination with the Fisher vector strategy. This shows that the time required for this video classification in this combination is somewhat negligible compared to the time it takes to build a visual dictionary

6.2.2 Impact of video sampling rate

In the video, before and after the adjacent video frames to a large extent, their content and information is about the same. Because in time, the extraction feature is the biggest calm in the video classification, so we try to experiment if we use different sampling rates to deal with video frames, then the accuracy of the classification results will be what kind of impact and hope that through this Way to improve the speed of the feature extraction process.

Our experimental process is as follows: If each two frames extract the image sequence, then our optical flow histogram feature block will contain 3 frames of image content; if every three frames extract the image sequence, then our optical flow histogram features The block will span 2 frames; if each of the 6 frames extracts the image sequence, then each optical stream histogram feature block contains only one frame of image content.

By changing the sampling frequency of the video frame, we find that the accuracy of the video classification result is greatly affected: for the Fisher vector method, when the sampling rate is 1, the accuracy rate is 90%; the sampling rate is 2, the accuracy rate is 89.4%; sampling rate of 6, the accuracy rate dropped to 87%. For hierarchical clustering, the effect of changing the sampling rate is not so obvious: the accuracy rate is 85.9% at the sampling rate of 1 and 84.2% at the sampling rate of 6. In terms of speed, it takes about 7.9 seconds to extract the video feature every frame, and the time is about 4.9 seconds for each two-frame sample. For each 6-frame sample, only 2.7 seconds is required. From the above results can be found, when using different video frame sampling rate, the calculation efficiency has been significantly affected. Therefore, we can consider the use of every 6 frames for frame sampling, so that the loss of 1% to 3% of the case, the calculation speed has increased by 2 seconds to 3 seconds.

6.2.3 Influence of optical flow algorithm

Prior to this, there was not much research on the effect of the selection of the optical flow on the characterization descriptor of the optical flow histogram. Therefore, we compare two different optical flow algorithms: Horn-Schunck algorithm and Lucas-Kanade algorithm in terms of accuracy and computational efficiency.

From the computational efficiency, the Lucas-Kanade algorithm takes 31 frames per second, slightly faster than the Horn-Schunck algorithm at 27 frames per second. In terms of accuracy, the Horn-Schunck algorithm has an accuracy of 90%

The Lucas-Kanade algorithm is 87%. This reflects the fact that the optical flow calculation method has a significant effect on the accuracy of the final recognition of the attitude recognition.

6.2.4 Selection of the best solution

Through our experiments, we can find that if you want to get the fastest or most accurate attitude recognition system, we recommend that you can use the following combination: histogram of optical flow - fisher vector - linear SVM

7. Project Management

7.1.Considerations

In order to guarantee the progress of this project, I need to take into account the budget of each step. the budget mainly consists of hardware and software.

7.2 Budget monitoring

Budget of step of the project will be updated in time considering the situation in order to control the total budget, and the final budget will be real and effective.

7.3 Hardware budget

a set of hardware will be needed as the basement of the project.

Product	Price	Unit
Macbook Pro	¥ 9500	1
Apple Iphone 7	¥ 6300	1
Apple Ipad	¥ 1800	1
Total Estimated	¥ 17600	

7.4 Software budget

Software products will be needed to carry out the project through all the progress. While some of them are available for free, others are cost.

Product	Price	Unit
Matlab	¥ 700	1
Xcode	¥ 0	1
Total	¥ 700	

7.5 Total budget

After adding all the budgets provided above, the total budget is showed as follow.

Concept	Price
Hardware	¥ 17600
Software	¥ 700
Total	¥ 18300

Conclusion

This paper mainly uses a kind of excellent performance and robust video characteristics: optical flow histogram characterization descriptor to express video data information. At the same time, in order to make the accuracy and computational efficiency of the system reach a high level, we evaluate and To find a kind of video classification for the most suitable for the word bag model combination.

Experiments show that, when the need to extract the optical flow histogram descriptor, every two frames to extract features than each frame are extracted features 1.5-1.7 times faster, but the loss of accuracy at this time can be ignored. When we are demanding the speed of the video classification system, it is possible to consider sampling the video for every six frames to obtain the optical flow histogram characterization descriptor. In addition, for the optical flow histogram descriptor, we find that the selection of the optical flow calculation method has a significant effect on the performance of the final system. Between the best performing Horn-Schunck algorithm and the second best Lucas-Kanade algorithm, the result is that there is a 5% gap.

For the construction of visual dictionary this step, the most accurate method is undoubtedly the Fisher method. Hierarchical k-means Hierarchical clustering takes up one-sixth of the time spent by the Fischer vector, but accordingly it sacrifices by a 2% reduction in accuracy. Therefore, if we want to get a more efficient video classification system, we can consider using Hierarchical k-means hierarchical clustering.

For support vector machine video classification, the use of cross-histogram kernel support vector machine in the performance of accuracy than the linear support vector

machine excellent. But the linear support vector machine is faster than the support vector machine of the crossed histogram kernel.

Acknowledgement

At the last semester of my college, my student career was spent in Barcelona, Spain. I would like to thank the teacher of the domestic Beihang Quyu Yu teacher, regardless of the pre-identified topics, mid-term update to confirm the progress of my graduation design guidance, take the trouble to answer my questions, answer doubts. Thanks also to Prof. Joan at the Catalan Polytechnic University, who regularly communicates with me to guide me in reporting on the process and content of graduation design and to make improvements to my program's problems and to my interest in exotic life.

Reference

Code

main.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% BOW pipeline: Gesture recognition using HOF
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Part 1: extract descriptors
% Part 2: Represent images by histograms of quantized features
% Part 3: classification
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
close all;

run('/Applications/MATLAB_R2015b.app/toolbox/vlfeat-0.9.20/toolbox/vl_setup');

%DATASET
```

```
dataset_dir = 'Gesture';

%FEATURES extraction method
%'Horn-Schunck'

desc_name = 'hofs';

%FLAGS
do_feat_extraction = 1;
do_split_sets = 1;

do_form_codebook = 1;
do_feat_quantization = 1;

do_svm_linar_classification = 0;
do_svm_intersection_classification = 1;




%PATH
basepath = '..';

wdir = pwd;
libsvmpath = [wdir(1:end-6) fullfile('lib','libsvm-3.11','matlab')];
addpath(libsvmpath);

%BOW PARAMETERS
nfeat_codebook = 130000;%number of descriptors used by k-means for the codebook generation
norm_bof_hist = 1;
```

```
% number of images selected for training
num_train_img = 8;
% number of images selected for test
num_test_img = 2;
% number of codewords (i.e. K for the k-means algorithm)
nwords_codebook = 50;

%image file extension
file_ext = 'avi';

%Create a new dataset split
file_split = 'split.mat';
if do_split_sets
    data = create_dataset_split_structure(fullfile(basepath, 'video', ...
        dataset_dir),num_train_img,num_test_img,file_ext);
    save(fullfile(basepath,'video',dataset_dir,file_split),'data')
else
    load(fullfile(basepath,'video',dataset_dir,file_split));
end
classes = {data.classname};%create cell array of class name string


%%%%%%%%%%%%%%%%%Part 1 %%%%%%%%%%%%%%%%%%%%%%
%%%%%

%% Load pre-computed HOF features for training images
% The resulting structure array 'desc' will contain one
% entry per images with the following fields:
%  desc.row = info.row;
%desc.col = info.col;
```

```
%desc.depth = info.depth;
%desc.hofs = hofs;
%desc.vidname = videoName;
%desc.Size = info.descSize;
lasti=1;
for i = 1:length(data)
    images_descs = get_descriptors_files(data,i,file_ext,desc_name,'train');
    for j = 1:length(images_descs)
                                                    fname =
fullfile(basepath,'video',dataset_dir,data(i).classname,images_descs{j});
        %fprintf('Loading %s \n',fname);
        tmp = load(fname,'-mat');
        tmp.desc.class=i;
       % tmp.desc.imgfname=regexprep(fname,['.' desc_name],'.jpg');
        desc_train(lasti)=tmp.desc;
        desc_train(lasti).hofs = single(desc_train(lasti).hofs);
        lasti=lasti+1;
    end;
end;

%% Load pre-computed HOF features for test images

lasti=1;
for i = 1:length(data)
    images_descs = get_descriptors_files(data,i,file_ext,desc_name,'test');
    for j = 1:length(images_descs)
                                                    fname =
fullfile(basepath,'video',dataset_dir,data(i).classname,images_descs{j});
        fprintf('Loading %s \n',fname);
        tmp = load(fname,'-mat');
        tmp.desc.class=i;
```

```
            %tmp.desc.imgfname=regexprep(fname,['.' desc_name],'.jpg');
            desc_test(lasti)=tmp.desc;
            desc_test(lasti).hofs = single(desc_test(lasti).hofs);
            lasti=lasti+1;
        end;
    end;

    %% Part2: Build visual vocabulary using k-means %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%

    if do_form_codebook
        fprintf('\nBuild visual vocabulary:\n');

        % concatenate all descriptors from all images into a n x d matrix
        DESC = [];
        labels_train = cat(1,desc_train.class);
        for i=1:length(data)
            desc_class = desc_train(labels_train==i);
            randimages = randperm(num_train_img);
            randimages = randimages(1:5);
            DESC = vertcat(DESC,desc_class(randimages).hofs);
        end

        % sample random M descriptors from all training descriptors
        r = randperm(size(DESC,1));
        %r = r(1:min(length(r),nfeat_codebook));

        DESC = DESC(r,:);
        numClusters = 5;
             %[centers, assignments] = vl_kmeans(DESC', numClusters,'Initialization',
'plusplus');
```

```
  % x = 1:length(assignments);

   data = DESC * 10^4 * 5;
   data = uint8(data');
   nleaves = 100;
   [tree,A] = vl_hikmeans(data,numClusters,nleaves);

   % run k-means
   VC = centers';
   clear DESC;
end

 %k-means descriptor quantization
 if do_feat_quantization
    fprintf('\nFeature quantization');
    quantdist = [];
    for i=1:length(desc_train)
       dmat = eucliddist(desc_train(i).hofs,VC);
       [mv,visword] = min(dmat,[],2);
       desc_train(i).visword = visword;
    end

    for i=1:length(desc_test)
       dmat = eucliddist(desc_test(i).hofs, VC);
       [mv,visword] = min(dmat,[],2);
       desc_test(i).visword = visword;
    end
 end
```

```
%% represent images with BOF histograms
%represent each image by the normalized histogram of visual
N = size(VC,1);

for i=1:length(desc_train)
    visword = desc_train(i).visword;
    edges = 1:1:numClusters;
    Hist = histcounts(visword,numClusters);
    Hist(1,:) = Hist(1,:)/norm(Hist(1,:),1);
    desc_train(i).bof = Hist;
end

for i=1:length(desc_test)
   visword = desc_test(i).visword;
   edges = 1:1:numClusters;
   Hist = histcounts(visword,numClusters);
   % normalize bow-hist (L1 norm)
   Hist(1,:) = Hist(1,:)/norm(Hist(1,:),1);
   % save histograms
   desc_test(i).bof = Hist;
end
```

```
%%%%%%%%%%%%%%%%%% Part 3: image classification %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Concatenate bof-histograms into training and test matrices
bof_train=cat(1,desc_train.bof);
bof_test=cat(1,desc_test.bof);
```

```
% Construct label Concatenate bof-histograms into training and test matrices
labels_train=cat(1,desc_train.class);
labels_test=cat(1,desc_test.class);



% LINEAR SVM
if do_svm_linar_classification
   % cross-validation
   C_vals = log2space(5,10,10);
   for i=1:length(C_vals);
      opt_string=['-t 0  -v 5 -c ' num2str(C_vals(i))];
      xval_acc(i)=svmtrain(labels_train,bof_train,opt_string);
   end
   %select the best C among C_vals and test your model on the testing set.
   [v,ind]=max(xval_acc);



   % train the model and test
   model=svmtrain(labels_train,bof_train,['-t 0 -c ' num2str(C_vals)]);
   disp('*** SVM - linear ***');
   svm_lab=svmpredict(labels_test,bof_test,model);

   method_name='SVM linear';
   % Compute classification accuracy
   compute_accuracy(data,labels_test,svm_lab,classes,method_name,desc_test,...
                    visualize_confmat & have_screen,...
                    visualize_res & have_screen);
end
```

```
% SVM with intersection kernel
if do_svm_intersection_classification

    %% train kernal
kernel_train = hist_isect(bof_train,bof_train);
kernel_train = [(1:size(kernel_train,1))',kernel_train];
bestc=200;bestg=2;
bestcv=0;
    %cross validation
    for log2c = -1:10,
       for log2g = -1:0.1:1.5;
          cmd = ['-v 9 -t 4 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
          %cv = svmtrain(labels_train, bof_train, cmd);
          cv = svmtrain(labels_train, kernel_train, cmd);
          if (cv >= bestcv)
             bestcv = cv;
             bestc = 2^log2c;
             bestg = 2^log2g;
          end
              fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv,
bestc, bestg, bestcv);
       end
    end

  options=sprintf('-s 0 -t 4 -c %f -b 1 -g %f',bestc,bestg);
  model=svmtrain(labels_train,kernel_train,options);

%% kernel test
kernel_test = hist_isect(bof_test,bof_train);
kernel_test = [(1:size(kernel_test,1))',kernel_test];
```

```
[predict_label, accuracy , dec_values] = svmpredict(labels_test,kernel_test, model,'-b 1');
end
```

Video2DenseHOFVolumes.m

```
function desc = Video2DenseHOFVolumes(videoName, video, blockSize, numBlocks, numOr, flowMethod)
% [hofs, info] = Video2DenseHOFVolumes(video, blockSize, numBlocks, numOr, flowMethod
%
% Get Densely Sampled Histogram of Oriented Optical Flow volume descriptors
% from a video. Final size of the volume per descriptor is
% (blockSize .* numBlocks) pixels.
% - Optical Flow is calculated using MATLABs CV toolbox
% - Soft cell-borders within a single frame (linear interpolation in
%   row and col directions)
% - Hard cell-borders in time direction
% - Sampling is as dense as subcells
%
% video:          N x M x F grayscale video
% blockSize:          1 x 3 vector with sub-block size in pixels ([pRow pCol pFrames]
% numBlocks:      1 x 3 vector with number of blocks [nRow nCol nZ]
% numOr (optional): Number of orientations for the histogram (default: 8)
% flowMethod(optional): Method of optical opticalFlow: {'Horn-
```

```
Schunck' (default), 'Lucas-Kanade'}
    %
    % hofs:            Hof descriptors
    % info:            Info structure containing e.g. coordinates
    %

    if nargin < 4
       numOr = 8; % Default: 8 orientations
    end

    if nargin < 5
       flowMethod = 'Horn-Schunck'; % Horn-Schunk optical opticalFlow
    end

    % Calculate optical flow from the video
    opticalFlow = Video2OpticalFlow(video, flowMethod);

    % Determine area of frame for which features can be extracted
    nR = size(opticalFlow,1); %
    nC = size(opticalFlow,2); %
    nF = size(opticalFlow,3); %
    extraPixelsR = mod(nR, blockSize(1));
    extraPixelsC = mod(nC, blockSize(2));
    extraFrames = mod(nF, blockSize(3));
    newR = nR - extraPixelsR;
    newC = nC - extraPixelsC;
    newF = nF - extraFrames;
    offsetR = floor(extraPixelsR / 2); % Shave borders if too many pixels
    offsetC = floor(extraPixelsC / 2);
    offsetF = floor(extraFrames / 2);
    rangeR = (1:newR) + offsetR; % Ranges determine final area of descriptor
```

```
extraction
    rangeC = (1:newC) + offsetC;
    rangeF = (1:newF) + offsetF;

    % Get matrices for doing multiplication to get subblocks (Real-time
    % Bag-of-Words paper, TMM, Uijlings 2010)
    subBlocksR = newR / blockSize(1);
    subBlocksC = newC / blockSize(2);
    subBlocksF = newF / blockSize(3);
    arrayA = DiagMatrixLinear(subBlocksR, newR);
    arrayB = DiagMatrixLinear(newC, subBlocksC);

    % Initialize block HOFs
    blockHof = cell(1, numOr);
    for i=1:numOr
       blockHof{i} = zeros(subBlocksR, subBlocksC, subBlocksF);
    end

    % Obtain subblocks
    frameBlockCount = 1;
    idxF = 1;
    for i=rangeF
       % Get oriented magnitudes of opticalFlow field for current frame
              omFrame = VectorField2D2OrientedMagnitude(numOr,
opticalFlow(rangeR,rangeC,i));

       % Aggregate responses over rows and columns and frames
       for j=1:size(omFrame,3)
          blockHof{j}(:,:,idxF) = blockHof{j}(:,:,idxF) + ...
                        arrayA * omFrame(:,:,j) * arrayB;
       end
```

```
    % Keep track if feature should be added to this frame or next
    % I.e. this keep track of how we sum over frames
    frameBlockCount = frameBlockCount + 1;
    if frameBlockCount > blockSize(3)
        idxF = idxF + 1;
        frameBlockCount =1;
    end
end

% Concatenate all orientations to get HOFs per subblock
theBlockHof = zeros(length(blockHof{1}(:)), numOr);
for i=1:numOr
    theBlockHof(:,i) = blockHof{i}(:);
end
clear blockHof

% Get info structure
info = GetDenseInfoStructure3D([subBlocksR   subBlocksC   subBlocksF], ...
                        [blockSize(1) blockSize(2) blockSize(3)], ...
                        [offsetR      offsetC      offsetF]);

% Determine which blocks need to be taken together
% Note that subBlocksR is the total number blocks found in this video
% while numBlock denotes the number of desired blocks per HOF descriptor
indices = 1:(subBlocksR * subBlocksC * subBlocksF); % all indices
indices = reshape(indices, subBlocksR, subBlocksC, subBlocksF); % indices
where subblock come from

% Now get coordinates for creating final features
totNumBlocks = numBlocks(1) * numBlocks(2) * numBlocks(3);
```

```
coordinates = cell(1, totNumBlocks);
idx = 1;
for k=1:numBlocks(3)
   for j=1:numBlocks(2)
      for i=1:numBlocks(1)
         coordinates{idx} = indices(i:end-numBlocks(1)+i, ...
                                    j:end-numBlocks(2)+j, ...
                                    k:end-numBlocks(3)+k);
         coordinates{idx} = coordinates{idx}(:);
         idx = idx + 1;
      end
   end
end
coordsFinal = cat(1, coordinates{:});

% Concatenate the features and reshape. The concatenation is done in such a
% way that the reshape gives the correct HOF features
hofs = theBlockHof(coordsFinal,:);
hofs = reshape(hofs, [], numOr * totNumBlocks);

% Update info structure by concatenating coordinates in the second
% dimension for each subcell (coordsFinal -> reshape) and then take the
% mean to get the middle coordinate of complete block
info.row = info.row(coordsFinal);
info.row = reshape(info.row, [], totNumBlocks);
info.row = mean(info.row, 2);

info.col = info.col(coordsFinal);
info.col = reshape(info.col, [], totNumBlocks);
info.col = mean(info.col, 2);
```

```
% Also add descriptor sizes to info structure
info.descSize = (blockSize .* numBlocks);

desc.row = info.row;
desc.col = info.col;
desc.hofs = hofs;
desc.vidname = videoName;
desc.Size = info.descSize;
```